

《深度逻辑》

統一 邏輯、深度學習、與人工智能

YKY

January 1, 2021

Summary

- BERT 在训练过程中「被逼」预测遮掩的词语，由此诱导出人类的知识，这已经具有 AGI 的雏形
- 从经典逻辑 AI 的角度看，BERT 内部可能有符合逻辑思维的结构，而我们很惊讶地发现，透过 Curry-Howard 对应，BERT 可以看成是一种另类的逻辑
- AGI 系统可以和 逻辑结构（以 范畴论、topos 表述）建立紧密的联系，这个联系可以指导往后的 AGI 发展路线，非常方便
- 我需要一些合作者帮助，特别是 BERT 和 soft Actor-Critic 方面

Contents

1 Structure of AI in the framework of reinforcement learning

An AI is essentially a dynamical system that constantly updates its “state” \boldsymbol{x} :

$$\dot{\boldsymbol{x}} = \boldsymbol{F}(\boldsymbol{x}) \quad (1)$$

Part of the state \mathbf{x} contains **sensory input** and **action output** that allow the AI to interact with the external environment.

2 Structure of logic

The central tenet of my theory is that the state \mathbf{x} of the AI system is consisted of **logic propositions** and that \mathbf{F} plays the role of the **logic consequence** operator \vdash :

$$\boxed{\text{propositions}} \vdash^{\mathbf{F}} \boxed{\text{propositions}} \quad (2)$$

So our goal now is to elucidate the structure of \vdash . Currently the most elegant formulation is given by **categorical logic** or **topos theory**.

我发觉 我是一个擅长於 “synthesize” 的人，意思是我会看很多书，然后将各种分散的 ideas 融合成一个 内部协调 的理论（当中大部分 ideas 不是我原创的）。

在接下来的篇幅，我会勾划一个 对於 AGI 来说是完整的 逻辑理论，而这理论 最中心的思想 是 Curry-Howard isomorphism....

3 Curry-Howard correspondence

Curry-Howard isomorphism 是一个很深刻的思想，如果不小心的话 甚至会觉得它讲了等於没讲。

它已经被发现了很多次,实际上它的发现者包括 : Brouwer-Heyting-Kolmogorov-Schönfinkel-Curry-Meredith-Kleene-Feys-Gödel-Läuchli-Kreisel-Tait-Lawvere-Howard-de Bruijn-Scott-Martin-Löf-Girard-Reynolds-Stenlund-Constable-Coquand-HuetLambek

Curry-Howard isomorphism 讲的是 **逻辑** 与 **计算** 之间的同构，但在 1990s Lambek 加上了 **category theory**，所以现在不少人会讲 Curry-Howard- Lambek. 事实上，逻辑-计算-范畴论 这个「三角关系」之间的相互作用非常丰富，人们认为是未来发展的思想泉源。

简单来说：当我们做 逻辑思考时，表面上有一种语法上 (syntax) 的形式，即 $A \Rightarrow B$ ：

$$\frac{\boxed{\text{logic}} \quad A \Rightarrow B}{\boxed{\text{program}} \quad \Box \overset{f}{\mapsto} \Box} \tag{3}$$

而在这 语法「底下」，还有一个 运算，它可以看成是执行 证明 (proof) 的工作，它将 A 的证明 map 到 B 的证明（为了避免符号累赘，我将这些 “proof witness” 都记作 \Box ，但它们每个是不同的）。

从另一角度看，Curry-Howard isomorphism 可以看成是 某些 状态 (states，例如 A) 和状态之间的 转换 (transitions，例如 $\overset{f}{\mapsto}$) 之间的 对偶。而这种 对偶 不断在 截然不同的范畴里出现：

logic	computation	category theory	physics	topology
proposition	type	object	system	manifold
proof	term	morphism	process	cobordism

(4)

前两个就是 Curry-Howard，第三个 是 Lambek 加上去的，其余的来自 John Baez & M. Stay 的论文： *Physics, Topology, Logic and Computation: a Rosetta stone* [2010]。例如在 physics 里面是 Hilbert space 和 operators 的对偶；在 topology 里面，cobordism 的著名例子就是这个 “pair of pants”：



In string theory，它表示上面的 strings 变成下面的 string 的「时间过程」。

3.1 Type theory

描述 program 或 computation 的语言叫 type theory. 例如在一般的 编程语言里可以有这样的一句：

$$\text{define length}(s: \text{String}): \text{Integer} \{ \dots \} \tag{6}$$

意思是说 length() 是一个函数，输入 String，输出 Integer.

在数学里 我们描述 函数 时会用：

$$f : A \rightarrow B \quad (7)$$

这个表达式其实就是 type theory 的一般形式：

$$\underbrace{\text{term}}_t : \underbrace{\text{type}}_T \quad (8)$$

而这个 notation $t : T$ 其实也可以写成 $t \in T$ （但不正统而已）。

换句话说，types 就是 **集合**，terms 是集合中的 **元素**。

更一般地，一个 type theory 的句子 可以包含 type context：

$$\underbrace{\text{context}}_{x : A} \vdash \underbrace{\text{type assignment}}_{f(x) : B} \quad (9)$$

意思就像在 program 的开头 “declare” 一些 变量 的类型，然后 program 就可以被 赋予 后面的 类型。

这个 \vdash 的过程 称为 type assignment，而这就是 type theory 做的全部工作。

λ -calculus

在一个 program 里，除了定义 类型，还需要定义 函数。这件工作是由 λ -calculus 负责。

λ -calculus 可以定义函数 而不需要提及它的「名字」。例如，用数学式表达：

$$f(x) \triangleq x^2 \quad (10)$$

它的 λ -表达式就是：

$$f \triangleq \lambda x. x^2 \quad (11)$$

注意：在 λ -表达式里，不需要提到 f 的「名字」。

λ -calculus 是由 Alonso Church 发明，目的是研究数学上 substitution 的性质。Substitute 是每个中学生都懂得做的事，但要用数学表达出来却是出奇地麻烦。

同时，Church 发现 λ -calculus 是一种「万有」的计算形式，和 Turing machines 等效。「AI 之父」John McCarthy 用 λ -calculus 发展出 Lisp 语言，它是所有 functional programming language 的鼻祖。

Curry-Howard correspondence

在 Curry-Howard 对应下, type A 就是 逻辑命题 A , type A 或 集合 A 里面的元素 是其 **证明** (proof, or proof witness)。

而, $A \Rightarrow B$ 也是 逻辑命题, 它对应於 the function type $A \rightarrow B$, 也可以写作 B^A , 而这个 type 或 集合 里面的 元素 就是一些 函数 $f : A \rightarrow B$. 如果 有一个这样的函数存在, 则 type $A \rightarrow B$ 有「住客」(inhabited), 换句话说 $A \Rightarrow B$ 有 **证明**。

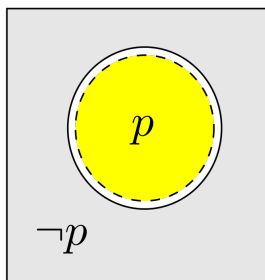
3.2 Intuitionistic logic

Curry-Howard isomorphism 揭示了 type theory 和 intuitionistic logic (直觉主义逻辑) 之间的关系。这种逻辑的特点是没有 **排中律** (law of excluded middle, LEM), 或者等价地, **double negation**, 即 $\neg\neg p \Rightarrow p$.

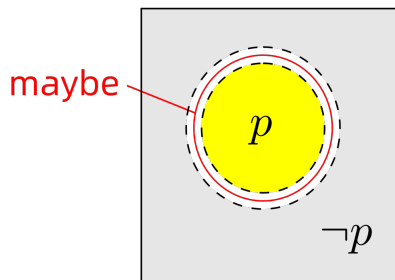
排中律 是说: $p \vee \neg p$ 是 恒真命题。但在 直觉主义 逻辑中, $p \vee \neg p$ 表示 p 的证明 **或** $\neg p$ 的证明, 但有时候这两者都不知道 (例如 现时仍未找到证明, 或者不可能找到证明)。

在 拓撲学 里, 一般用 **open sets** 表示空间中的子集 (这习惯起源自 Hausdorff 时期)。但 p 的 **补集** \bar{p} 并不 open, 所以要将 $\neg p$ 定义为 p 的补集的 **interior**, 即 $\neg p \triangleq \bar{p}^\circ$. 於是 $p \cup \neg p \neq \text{Universe}$: *

classical logic



intuitionistic logic



(12)

附带一提: 人们惊讶地发现, 在直觉主义逻辑下, axiom of choice \Rightarrow law of excluded middle. 换句话说, axiom of choice 和 直觉主义 也有内在的矛盾。

* Diagram from the book: *Classical and Non-classical Logics – an introduction to the mathematics of propositions* [Eric Schechter 2005], p.126.

3.3 Higher-order logic

Propositional logic 的意思是：只有命题，但忽略任何 **命题内部** 的结构。

假设 p, q 是命题，命题逻辑的基本运算 就是 $p \wedge q, p \vee q, p \Rightarrow q, \neg p$.

First-order logic 的意思是：容许 这样的方法 构成 命题：

$$\overbrace{\text{IsHuman}}^{\text{predicate}}(\overbrace{\text{John}}^{\text{object}}). \quad (13)$$

Predicate 的意思是 **谓词**；谓词 是一些「有洞的命题」，它们被填入 objects 之后就变成完整的命题。类似地可以有 **多元**的 predicates，例如：

$$\text{Loves}(\text{John}, \text{Mary}). \quad (14)$$

First-order 指的是： \forall, \exists 这些 **量词** 可以 **作用** 在 objects 的类别上，例如（Mary 人见人爱）：

$$\forall x. \text{Loves}(x, \text{Mary}) \quad (15)$$

但 first-order logic 不容许 量词 作用在 predicates 的类别上，除非用 second-order logic.

一个 二阶逻辑的例子是「拿破仑 具有一个好将军应该具备的所有特质」：

$$\forall p. p(\text{Good General}) \Rightarrow p(\text{Napoleon}). \quad (16)$$

注意 p 是在 predicates 的类别之上量化的。

3.4 旧式 logic with type theory

Type theory 的历史还可以追溯更早。它起源於 Russell 为了解决 **逻辑悖论**，例如：「一个只帮自己不理发的人理发的理发师帮不帮自己理发？」这些 逻辑悖论 根源是在於：定义一样东西的时候，中途 **指涉** 了这个东西本身。这种不良的定义称作 **impredicative**. 为了避免不良定义，每个东西出现之前必需「宣告」它的类型，这就是 type theory 原来的目的。

在 Curry-Howard isomorphism 未被重视之前，有一种更简单地 用 type theory 定义 逻辑的方法。在这种方法下，逻辑命题 $p, q, p \wedge q$ 等 **直接用** terms 定义，而不是像 Curry-Howard 那样，逻辑命题 = types，证明 = terms.

在这情况下 type theory 处理的是 (first- or higher-order) predicate logic 的方面。

3.5 Martin-Löf type theory

Per Martin-Löf was the first logician to see the full importance of the connection between intuitionistic logic and type theory.

3.6 Arithmetic-logic correspondence

很多人都知道，经典逻辑中 \wedge, \vee 对应於 **算术运算** $\times, +$ （也可以看成是 fuzzy logic 的 \min, \max 。）其实这就是 George Boole 尝试将 **逻辑** 变成 某种**代数** 的原因。

较少人知道的是 $A \Rightarrow B$ 也对应於 B^A ：

A	B	$A \Rightarrow B$	B^A
0	0	1	$0^0 = 1$
0	1	1	$1^0 = 1$
1	0	0	$0^1 = 0$
1	1	1	$1^1 = 1$

(17)

其中 0^0 是「不确定式」，但根据 组合学 惯例可以定义为 1.

这个惊奇的「巧合」似乎再一次证实 Curry-Howard correspondence 是正确的。

3.7 Internal language and classifying topos

We have the following transformations between two formalisms:

$$\boxed{\text{topos}} \mathcal{C} \begin{array}{c} \xrightarrow{\text{internal language}} \\ \xleftarrow{\text{classifying topos}} \end{array} T \boxed{\text{type theory}} . \quad (18)$$

In other words,

$$\mathcal{C} = \text{Cl}(T), \quad T = \text{Th}(\mathcal{C}). \quad (19)$$

4 Intuitionistic logic

不要忘记 Gödel's interpretation of intuitionistic logic using possible-world semantics!

In topos theory $A \Rightarrow B$ is adjoint (via the hom-product adjunction) to $A \vdash B$, which is “okay” because it is independent of which implication (material or strict) we are using.

In a topos \mathbb{E} , the subobject $\text{Sub}_{\mathbb{E}}(A)$ is a **poset** that admits **Heyting implication**. The Heyting implication $a \Rightarrow b$ exists for all elements a, b, x such that:

$$x \leq (a \Rightarrow b) \quad \text{iff} \quad (x \wedge a) \leq b. \quad (20)$$

Every Boolean algebra can be a Heyting algebra with the material implication defined as usual: $a \Rightarrow b \equiv \neg a \vee b$.

Heyting algebra is to intuitionistic logic what Boolean algebra is to classical logic. But this may not jibe with the idea of “strict implication”.

Under Kripke semantics, the Heyting arrow \rightarrow can be defined by:

$$k \Vdash A \rightarrow B \quad \Leftrightarrow \quad \forall \ell \geq k \ (\ell \Vdash A \Rightarrow \ell \Vdash B) \quad (21)$$

Whereas the “fish-hook” strict implication can be defined as:

$$A \multimap B \quad \equiv \quad \Box(A \Rightarrow B) \quad (22)$$

The two can be regarded as equivalent via:

$$\begin{aligned} k \Vdash \Box(A \Rightarrow B) &\Leftrightarrow \forall \ell \geq k \ (\ell \Vdash (A \Rightarrow B)) \\ &\Leftrightarrow \forall \ell \geq k \ (\ell \Vdash A \Rightarrow \ell \Vdash B) \end{aligned} \quad (23)$$

5 The problem of “material implication”

Material implication 的意思是「实质蕴涵」，亦即是说 $A \Rightarrow B$ 等价於 $\neg A \vee B$ ，其真值表如下：

A	B	$A \Rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

(24)

Material implication 的概念向来很有争议，例如，当前提是错误时，它永远是真的：

$$\text{瑞士在非洲} \Rightarrow \text{猪会飞} \quad (25)$$

透过观察 truth table 可以发现，它的每一列 其实代表一个 可能世界，这些 可能世界 是不会 同时发生的。换句话说，material implication 和 strict implication 本来是一样的，只是前者将 可能世界 的语义 隐蔽到「幕后」。

For strict implication to make sense, it is always necessary to invoke possible-world semantics. A strict implication is always **learned** from numerous examples from experience, in accord with the philosophical tradition of “empiricism”.

Strict implication is equivalent to material implication over multiple instances. The truth table of material implication agrees with the functional interpretation of implication.

6 \forall and \exists as adjunctions

Let $\text{Forms}(\vec{x})$ denote the set of formulas with only the variables \vec{x} free.

Then there is a trivial operation of adding an additional dummy variable y :

$$* : \text{Forms}(\vec{x}) \rightarrow \text{Forms}(\vec{x}, y) \quad (26)$$

taking each formula $\phi(\vec{x})$ to itself.

It turns out that \exists and \forall are adjoints to the map $*$:

$$\exists \dashv * \dashv \forall \quad (27)$$

7 Sheaves and topos

Some **Set**-valued functors are **representable**, ie, isomorphic to a hom-functor.

Functors $\mathcal{C} \rightarrow \mathbf{Set}$ are called **pre-sheaves** on \mathcal{C} .

Sheaves capture “indexing”.

7.1 Yoneda lemma

How sheaves gives rise to representables.

8 Semantics

8.1 Model theory / functorial semantics

8.2 Topological interpretation (intuitionistic logic)

8.3 Generalized elements and forcing

一个 逻辑命题 ϕ 可以看成是由 某论域 $A \xrightarrow{\phi} \Omega$ 的函数, 其中 $\Omega = \{\top, \perp\}$.

也可以说: 命题 $\phi(x)$ 是真的, 其中 x 是 A 的**元素**。In category theory, we use the terminal object 1 to “pick out” elements of A , as follows:

$$1 \xrightarrow{x} A \xrightarrow{\phi} \Omega. \quad (28)$$

通常, 任意一个由 1 出发的函数 $x: 1 \rightarrow A$ 可以直接看成是 A 的「元素」。

但如果我们用另一个论域 C 取代 1, 换句话说:

$$C \xrightarrow{x} A \xrightarrow{\phi} \Omega. \quad (29)$$

这样的 $x : C \rightarrow A$ 叫作 A 的 **generalized element**.

另一个术语是: C **forces** $\phi(x)$, notation $C \Vdash \phi(x)$.

或者说 $\phi(x)$ is true **at stage** C (这术语来自 possible-world semantics) .

8.4 Kripke-Joyal semantics

Cohen's (dis)proof of Continuum Hypothesis

Continuum hypothesis (CH):

$$2^{\aleph_0} = \aleph_1 \quad (30)$$

这是说: 连续统 $[0, 1]$ 的基数 2^{\aleph_0} 紧接在 可数集合的基数 之后。

1878 年, Cantor 提出 CH

1900 年, Hilbert 列出 连续统假设 为「23 问题」的第一个

Hilbert 给出了一个证明, 但里面有 bug

1938 年, Gödel 证明 $ZF + CH$ is consistent, 换句话说: ZF cannot disprove CH

1963 年, Paul Cohen 证明 ZF cannot prove CH

他用的方法叫 “forcing”

8.5 Kleene realizability

9 Homotopy type theory

9.1 What is homotopy?

9.2 Univalence axiom

10 Fuzzy logic

首先是 implication 的问题, fuzzy implication 并不对应於 material implication in Boolean algebra.

另外有个问题就是要考察一下 fuzzy truth value 在各种情况下的正确性。

例如假设 set 里面有 fuzzy proposition 的「证明」

又或者「人类」的集合是「有人类」这个命题的证明

而,「数学家」作为「人类」的子集,等於命题「所有人都是数学家」的证明

而这和 fuzzy value 是一致的

但为什么「John 是人」这个命题有点怪怪的?

如果它有 fuzzy value, 应该是某集合的子集

有些元素证明 John 是人, 有些证明他不是人

或者是 John 的属性的集合?

而其中有些属性 imply 他是人?

或者有些属性 \subseteq 人的属性?

还有这跟 “Marilyn Monroe is sexy” 是不是一致? Marilyn 的所有属性集合, 其中 imply sexy 的 subset

还是 sexy 的所有属性集合, 其中 Marilyn 也有的?

Sexy(marilyn), Human(john), vs Human(Mathematicians).

What kind of mapping does this require?

10.1 Fuzzy implication

Implication 能不能 generalize 到 fuzzy logic 的情况?

10.2 Fuzzy functions?

What are fuzzy functions?

11 Modal logic

A modal operator (such as \Box) in $\text{Sheaf}(X)$ is a sheaf morphism $\Box : \Omega \rightarrow \Omega$ satisfying 3 conditions, for all $U \subseteq X$ and $p, q \in \Omega(U)$:

- a) $p \leq \Box(p)$
 - b) $(\Box; \Box)(p) \leq \Box(p)$
 - c) $\Box(p \wedge q) = \Box(p) \wedge \Box(q)$
- (31)

11.1 Possible-world semantics

Possible-world semantics is also called **intensional semantics**, as opposed to **extensional semantics** where truth values are directly assigned to propositions. Does this idea jibe with the other definition of “intension”, ie, as opposed to Leibniz extensionality and also related to intensional logic?

要在电脑上实现 possible-world semantics 是不是很麻烦？

11.2 Intensional vs extensional

“Beethoven’s 9th symphony” and “Beethoven’s choral symphony” has the same **extension** but different **intensions**.

11.3 Intensional logic

Possible-world semantics is also called **intensional semantics**, as opposed to **extensional semantics** where truth values are directly assigned to propositions.

Logic terms differ in intension if and only if it is **possible** for them to differ in extension. Thus, **intensional logic** interpret its terms using possible-world semantics.

11.4 Symmetry in logic

- 词语 组成 句子，类比於 逻辑中，概念 组成 逻辑命题
- 抽象地说，逻辑语言 可以看成是一种有 2 个运算的 **代数结构**：加法（ \wedge ，合并命题，可交换）和 乘法（ \cdot ，用作概念合成，不可交换）
- 例如：

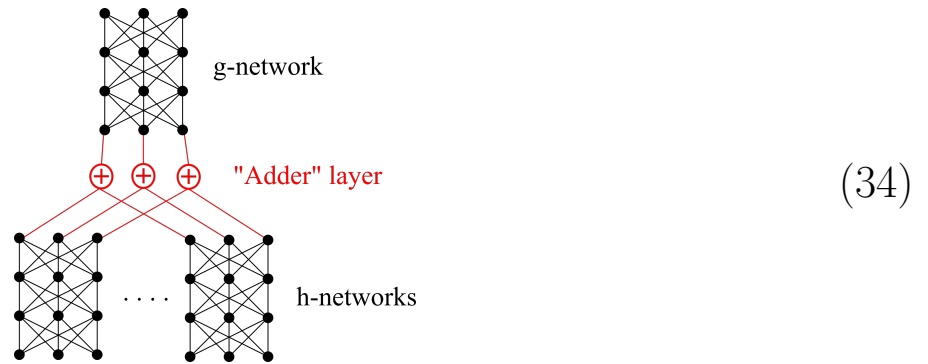
$$\begin{aligned} A \wedge B &\equiv B \wedge A \\ \text{下雨} \wedge \text{失恋} &\equiv \text{失恋} \wedge \text{下雨} \end{aligned} \tag{32}$$

- Word2Vec 也是革命性的；由 Word2Vec 演变成 Sentence2Vec 则比较容易，基本上只是 向量的 **合并** (concatenation); Sentence 对应於 逻辑命题
- 但 命题的 **集合** 需要用 symmetric NN 处理，因为 集合的元素 是**顺序无关**的

11.5 Symmetric neural network

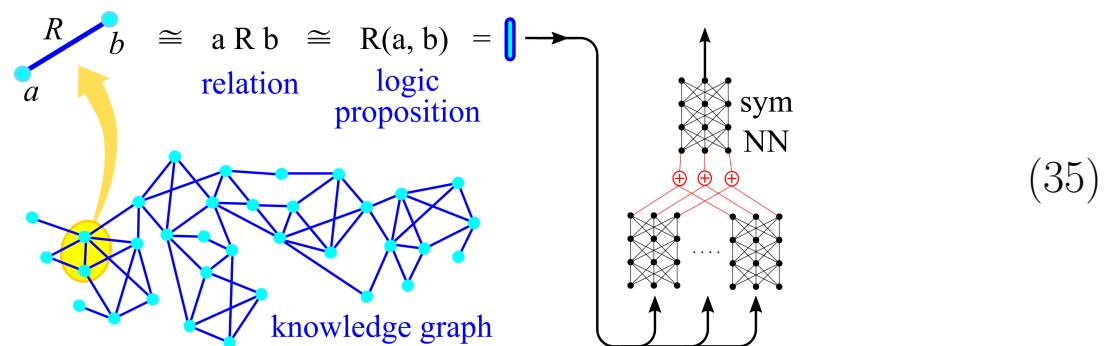
- Symmetric NN 问题 已经由 两篇论文解决了：
[PointNet 2017] [DeepSets 2017]
- Any symmetric function can be represented by the following form (a special case of the Kolmogorov-Arnold representation of functions):

$$f(x, y, \dots) = g(h(x) + h(y) + \dots) \quad (33)$$



11.6 知识图谱 (knowledge graphs)

- 知识图谱 不能直接输入神经网络，它必需分拆成很多 edges，每个 edge 是一个 关系，也是一个 逻辑命题；也可以说 “graphs are isomorphic to logic”



- 而这些 edges 似乎必需用 symmetric NN 处理，因为它们是 permutation invariant
- 逻辑化 建立了 知识图谱 和 BERT 之间的一道桥梁
接下来讨论 BERT....

11.7 逻辑与 AI 之间的联系

- 既然 AI 基於 逻辑，则 AI 与逻辑之间 必然存在 精确 (precise) 的联系
- BERT 似乎是在执行 句子之间的变换，而这些句子是 word embedding 的 concatenation，例如：

$$\text{苏格拉底} \cdot \text{是} \cdot \text{人} \xrightarrow{BERT} \text{苏格拉底} \cdot \text{会} \cdot \text{死} \quad (36)$$

这个做法看似很「粗暴」，其实它和 逻辑式子 的作用一样：

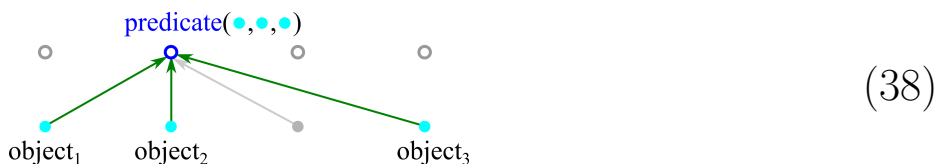
$$\forall x. \text{Human}(x) \Rightarrow \text{Mortal}(x) \quad (37)$$

而这式子，根据 Curry-Howard 对应，就是 (??) 的函数映射！

- 我会在另一辑 slides 里 简介一下这些理论；可以说，逻辑的 几何结构 是「永恒」的，它可以指示 AI 的 长远发展

11.8 谓词 (predicates) vs 命题 (propositions)

- “Predicate” 来自拉丁文「断言」的意思
- 逻辑里，predicate 代表一个 没有主体 / 客体 的断言，换句话说，是一个有「洞」的命题
- 命题 = 谓词 (predicate) + 主体 / 客体 (统称 objects)
- 例如：Human(John), Loves(John, Mary)
- 从逻辑的角度看，attention 的输出可以看成是 predicate 和 objects 的 结合：



- 形象地说：

$$\begin{aligned} \text{predicate} + \text{objects} &= \text{proposition} \\ \text{blue circle} + \text{blue dots} \dots &= \text{blue line} \end{aligned} \quad (39)$$

11.9 对 逻辑主义 的质疑

- 很多人怀疑：人脑真的用 逻辑 思考吗？
- 其实我们每句表达的 语言，都是逻辑形式的 (logical form)
- 直觉认为，人脑 构造一些 models，再从 model 中「读出」一些结论
- 例如给定一个描述：「已婚妇人出轨，用刀刺死丈夫」



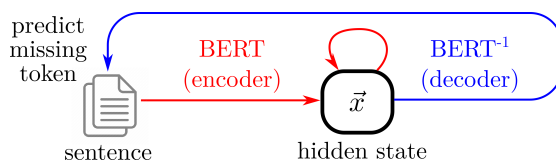
(40)

- 那么 妻子穿著什么衣服？衣服什么颜色？这些都是 臆想 出来的细节，是不正确的
- 这个 model 可以有哪些细节？答案是：任何细节都不可以有，除非是 逻辑上蕴含的，或被 逻辑约束
- Model 本身可以是一些 抽象的逻辑命题 构成的，这也合理；反而，一个有很多感官细节的 model 并不合理
- 其实人脑可能 比我们想像中 更接近逻辑

12 BERT as an alternative logic

12.1 BERT 的革命性意义：「闭环路训练」

- BERT 利用平常的文本 induce 出知识，而这 representation 具有 通用性 (universality)：



(41)

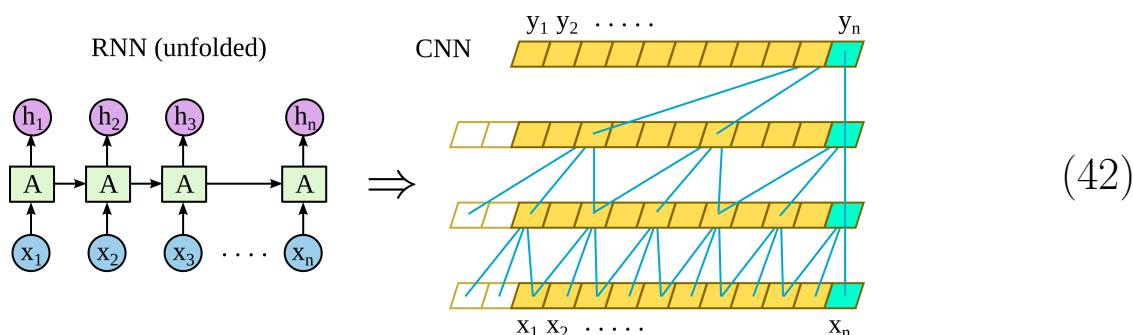
换句话说：隐状态的 representation 压缩了句子的意思，而它可以应用在别的场景下

- This implies that human-level AI can be *induced* from existing corpora, 而不需重复 像人类婴儿成长的学习阶段
- 这种训练方法是较早的另一篇论文提出，它并不属于 BERT 的内部结构

12.2 BERT 的内部结构

其实，BERT 也是混合了很多技巧 发展而成的：

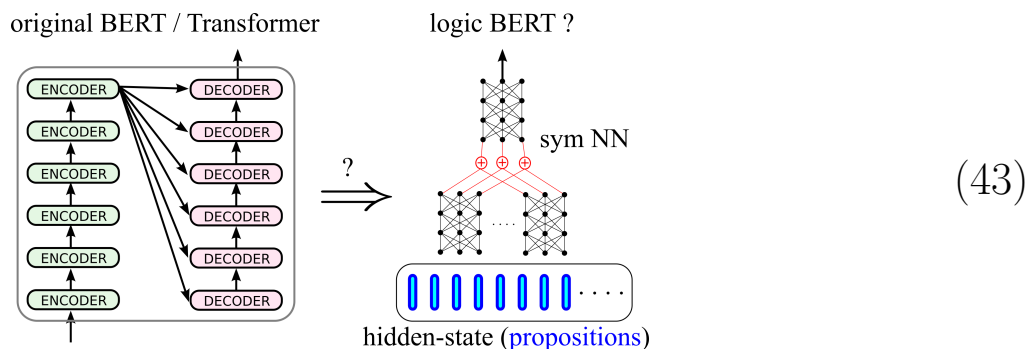
- BERT 基本上是一个 seq-to-seq 的运算过程
- Seq-to-seq 问题最初是用 RNN 解决的
- 但 RNN 速度较慢，有人提出用 CNN 取代：



- CNN 加上 attention mechanism 变成 Transformer
- 我的想法是重复这个思路，但引入 逻辑的对称性

12.3 BERT 的逻辑化

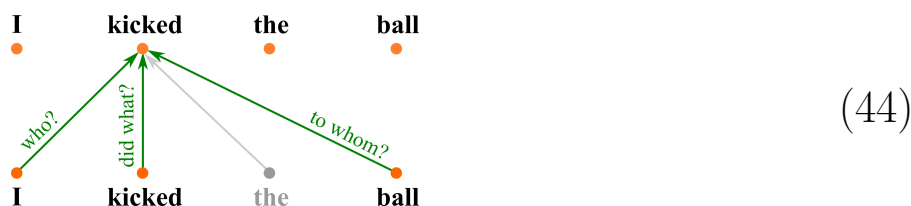
- 可以强迫 BERT 的 隐状态 变成 “set of propositions” 的形式，方法是将 对称性 施加在 Encoder 上：



- 下面会看到，BERT 的「注意力机制」已经是对称的，它可以做 逻辑推导

12.4 Attention 是什么？

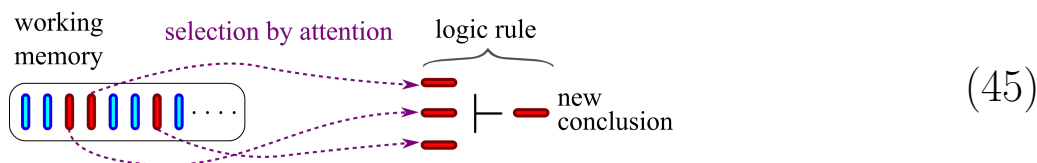
- 注意力 最初起源於 Seq2seq, 后来 BERT 引入 self-attention
- Attention 的本质就是 加权, 权值 可以反映 模型 关注的点
- For each input, attention weighs the **relevance** of every other input and draws information from them accordingly to produce the output
- 在 BERT 里, attention 是一种 **words** 之间的关系:



- 但, 从逻辑的角度看, word \neq 命题
- 在逻辑学上, 必需分清 命题内部 与 命题之间 这两个层次, 非常关键!

12.5 “Attention is all you need” ?

- 类似地, 高层的 attention 可以处理 命题之间 的关系
- 我们希望 attention 做到的是 选择有关联的命题, 去做逻辑推导:

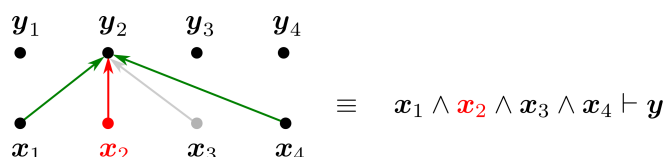


- 但从 N 个命题中选择 K 个, 可以有 $\binom{N}{K}$ 个子集, 是 exponential 的
- BERT 的做法是: 每次只输出 N 个命题, 而前提的 “support” 也是上一层的 全部 N 个命题, 每个前提的「影响力」由 matrix 权重决定
- 根据 Curry-Howard isomorphism, BERT 的映射 其实对应於某种 另类的逻辑 (BERT 的设计者可能也意识到这点), 这种逻辑的好处是运行非常快

- BERT 的逻辑 看似有局限，但这种表面的局限未必阻止它是 universal 的逻辑
- 关键是在 速度 与 逻辑的 expressive power 之间 找到平衡
- 最简单的 attention 公式是（其中 Q, K, V = query, key, value 矩阵）：

$$\mathbf{y}_j = \sum_i \langle Q \mathbf{x}_j, K \mathbf{x}_i \rangle V \mathbf{x}_i \quad (46)$$

（红色 代表注意力的 focus）这对应於一个逻辑式子：



$$\equiv \mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \mathbf{x}_3 \wedge \mathbf{x}_4 \vdash \mathbf{y}_2 \quad (47)$$

亦即是说： \mathbf{y}_j 是由 $\mathbf{x}_1, \dots, \mathbf{x}_n$ 得出的逻辑结论 with focus on \mathbf{x}_j

- 所谓 “focus” 并不是 逻辑概念，它只是 BERT 加速的 heuristic
- 容易看到，attention 对於 $\mathbf{x}_1, \dots, \mathbf{x}_n$ 是 交换不变的 (equivariant)，这表示每层 attention 的输出是一些 逻辑命题，和我的理论相符
- Multi-head attention 亦有一个很好的逻辑解释：即使 focus = \mathbf{x}_j ，亦有其他不同的前提，可以导致不同的结论，例如：

$$\mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \mathbf{x}_3 \vdash \mathbf{y}_1 \quad , \quad \mathbf{x}_1 \wedge \mathbf{x}_4 \wedge \mathbf{x}_5 \vdash \mathbf{y}_2 \quad (48)$$

12.6 BERT 为什么成功？

- 6 层的 BERT 有 $512 \times 6 = 3072$ 个 head，如果 $8 \times$ multi-head 则有 24576 个
- Each head does not simply correspond to 1 formula in conventional logic, may require further in-depth analysis....
- 我的猜测是：BERT 的高层 representation 是一些有 “high-level” 意义的命题，就像在视觉中，高层特征 代表一些复杂的物体

- The embedding of high-level propositions in vector space may be “semantically dense”, meaning that slight changes in the vector position may convey many different meanings
- 由於 logic rules 是以 6 层的 hierarchy 组织而成，这結構具有 “deep learning” 的特性

12.7 改良 BERT: 类似 attention 的方法

- The BERT attention formula (??) has some unnecessary restrictions, where generally we just need a symmetric function in the \mathbf{x}_i 's
- The general form of symmetric functions is given by (??)
- Immitating BERT, we introduce a “focus” of attention on \mathbf{x}_j :

$$\mathbf{y}_j = g(h(\mathbf{x}_j, \mathbf{x}_1) + \dots + h(\mathbf{x}_j, \mathbf{x}_n)) \quad (49)$$

this preserves **equivariance**

- We can use this function to replace the entire BERT Encoder:



12.8 逆因推理 (abduction) 与 自然语言理解

- 根据 Jerry Hobbs 的 “abductive interpretation of natural language” 理论，语言理解 是一个 **解释** (explain) 的过程
- **解释** 在逻辑上等同於 **逆因推理** (abduction)，亦即是 **逻辑蕴涵** (implication, $A \Rightarrow B$) 的反方向
- 举例来说：天气热 \Rightarrow 流汗，所以「天气热」就是「流汗」的 **解释**

- 这个理论 今天很少人知道，因为属于 经典逻辑 AI 时期

2013 年 他
获得
计算语言学
终身成就奖



12.9 捕捉更广泛的 semantics

- 阅读时，我们（人脑）有时可以预测下个 word（这是 BERT 训练的目标），但有时即使不能预测，但看到 next word 之后，仍会有某种「意料之内」的感觉
- 举例来说：

「天气热，我不停 流汗」

「天气热，我不停 吃冰淇淋」

第二个例子是比较少见的，但也合理

- （从逻辑角度看）BERT 只会预测 **最有可能** 的结论：

$$\text{前提} \xrightarrow{BERT} \text{预测} \quad (51)$$

其实我们想要的是：

$$\text{前提} \longrightarrow \text{很多不同的 预测} \quad (52)$$

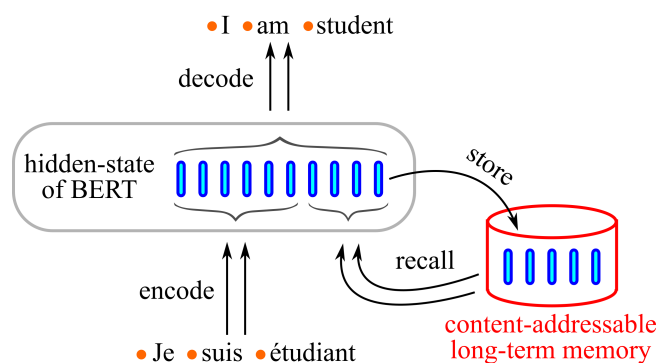
而只要其中一个预测成功，即给予「奖励」

- 这个情况和 **强化学习** 中的“stochastic actions”类似，需要的是：stochastic, multi-modal, continuous actions （例如 SAC, soft actor-critic）

12.10 Content-addressable long-term memory

- Content-addressable memory 的想法来自 Alex Graves *et al* 的 Neural Turing Machine [2014]

- 以前 BERT 的隐状态 没有逻辑结构，我们不是很清楚它的内容是什么；逻辑化之后，BERT 内部的命题可以储存在 长期记忆 中：



(53)

- 训练的方法，可以将所有 **—** 存进 memory，如果对结论有帮助则加分，错则减分
- 命名：Knowledge-Enhanced Reasoning with Memorized Items
- 这种系统 已非常接近 strong AI，而这是有赖 逻辑化 才能做到的
- 例如：「太阳是热的」、「水向下流」是经常正确的命题
- 但这些知识很多是 implicitly 存在於 rules (matrix weights) 之中
- 也有些知识是 explicit 的，例如：「猫是哺乳类动物」、「吸烟可以致癌」
- 逻辑化理论提供一种 诠释 logic rules (weights) 的方法
- 也可以将 weights 存进 content-addressable memory
- 但这个 idea 暂时仍未成熟，有待更多研究

13 References

欢迎提问和讨论 ☺

References

- [1] Alex Graves, Greg Wayne, and Ivo Danihelka. “Neural Turing Machines”. In: *CoRR* abs/1410.5401 (2014). arXiv: 1410.5401. URL: <http://arxiv.org/abs/1410.5401>.
- [2] Qi et al. “Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *CVPR* (2017). <https://arxiv.org/abs/1612.00593>.
- [3] Zaheer et al. “Deep sets”. In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 3391–3401.